617244

SOFIA's Choice: Scheduling Observations for an Airborne Observatory

Jeremy Frank, Elif Kurklu*
Computational Sciences Division
NASA Ames Research Center, MS 269-2
{frank,ekurklu}@email.arc.nasa.gov
Moffett Field, CA 94035

Abstract

We describe the problem of scheduling observations for an airborne observatory. The problem is more complex than traditional scheduling problems in that it incorporates complex constraints relating the feasibility of an astronomical observation to the position and time of a mobile observatory, as well as traditional temporal constraints and optimization criteria. We describe the problem, its proposed solution and the empirical validation of that solution.

Keywords: Planning, scheduling, stochastic search, constraint satisfaction, astronomy, flight planning

Introduction

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is NASA's next generation airborne astronomical observatory. The facility consists of a 747-SP modified to accommodate a 2.7 meter telescope. Employing a suite of optical, infrared, and sub-millimeter instrumentation, the observatory spans operational wavelengths of 0.3 to 1600 microns. SOFIA is expected to fly an average of 140 science flights/year over it's 20 year life time, double the previous rate of the KAO. The SOFIA telescope is mounted aft of the wings on the port side of the aircraft and is articulated through a range of 20 to 60 degrees of elevation. The telescope has limited lateral flexibility; thus, the aircraft must turn constantly to maintain the telescope's focus on an object during observations. Most flights will originate and terminate at Moffett Field, CA; therefore, it is necessary for the observatory flight plans close in on themselves. This typically requires an astronomical observing plan covering both Galactic and extra-galactic targets.

A significant problem in future SOFIA operations is that of scheduling Facility Instrument (FI) flights in support of the SOFIA General Investigator (GI) program. GIs are expected to propose single observations, and many observations must be grouped together to make up single flights. The SOFIA science staff is expected to have 3 - 4 facility science instruments to support GIs. Approximately 70 GI flight per year are expected, with 5-15 observations per flight. The scope of the flight planning problem for supporting GI observations with the anticipated flight rate for SOFIA makes the manual approach for flight planning daunting. There has been considerable success in automating the scheduling of jobs in a wide variety of industries with many different types of constraints. In this document, we describe the application of automated scheduling techniques to this problem.

*QSS Group Inc.

Problem Description

In this section we describe the details of what we call the *Single Flight Planning Problem* (SFPP). We divide the description into two components. We first describe the rules of the SFPP domain, and then we describe the inputs that are needed to define an instance of the SFPP.

Inputs and Objective

The scope of the problem we consider here assumes that a number of individual observation requests from GIs have been rated and accepted for observing. We then consider the problem of attempting to schedule a subset of these observations on a single flight. The input to the SFPP is defined as follows:

- A set of observation requests. Each request consists of
  - an object's Right Ascension (RA) and Declination (Dec)
  - its observation duration.
  - its priority.
  - its earliest start time and latest end time.
- A flight date.
- A flight duration.
- A flight horizon (i.e. earliest takeoff time and latest landing time).
- A designated takeoff and landing airport (which need not be the same.)

The objective is to find a flight plan that maximizes the summed priority of the observations of the observations performed. Since it is intractable to find the best possible plan, we will relax this constraint and limit ourselves to searching for *good* plans that perform many observations of high priority.

The Single Flight Planning Domain

The principal constraints that govern the SFPP is that which governs when a particular observation can be viewed from the aircraft, and how the aircraft's position evolves over the course of a flight. The first important point is the equation that governs the elevation of an object given its coordinates and the position and time at which the observation is performed.

We will follow the conventions in Meeus (Meeus 1991) in defining the visibility conditions for an astronomical object and the equations of motion of the aircraft that follow. Meeus uses the following definitions:

- α is the RA of the object to be observed.
- δ is the Dec of the object to be observed.
- ϕ is the Earth latitude at which the observation occurs. Positive latitudes are in the Northern hemisphere.
- L is the Earth longitude at which the observation occurs. Positive longitudes are measured West from Greenwich.
- θ is the "time" [1] at which the observation is performed.
- h is the elevation angle of the object relative to the horizon at the location of the observation. Positive elevations are above the horizon.
- A is the azimuth of the observation at the location of the observation. Azimuth is measured in degrees West, assuming that 0 degrees is due South.

Suppose all quantities except h and A are fixed. We can solve for h and A as follows:

$$H = \theta - L - \alpha \tag{1}$$

$$\sin h = \sin\theta \sin\delta + \cos\theta \cos\delta \cos H \tag{2}$$

$$\tan A = \frac{\sin H}{\cos H \sin\theta - \tan\delta \cos\theta} \tag{3}$$

As mentioned previously, since the telescope has minimum and maximum elevation limits, there may be times when an object is not visible. Figure 1 shows a plot that shows when an object is within and outside the telescope's elevation limits as time passes.

Also mentioned previously, the aircraft must turn continuously during an observation in order to keep the object in view during an observation. Figure 1 also shows the direction that the aircraft must fly in order to observe the object over time, at a fixed position. During an observation, the aircraft's ground track will be a curve. The consequence is that the position of the aircraft after completing an activity is a complex function of the object's coordinates and the start time. Beginning with the previous equations describing the azimuth of the observation, the equations of motion of the aircraft can be derived. Suppose the *ground speed* V of the

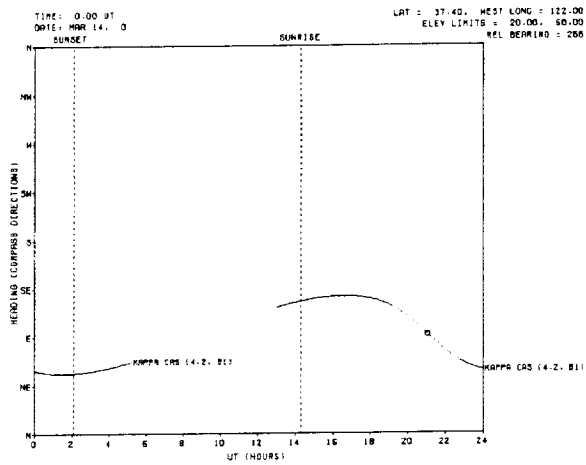

Figure 1: Visibility and azimuth of an observation performed over Moffett Field, CA. as a function of time. The gap in the graph indicates that the object elevation is below 20 degrees, and the dotted line indicates that the object elevation is above 60 degrees. Notice how the azimuth can vary greatly over periods of time as short as half an hour.

[1] Time, in this case, refers to Geenwich Sidereal Time, which is related to how long it takes the Earth to orbit the Sun.

aircraft is known and constant, the *relative bearing* [2] of the telescope is r, and that the Earth is a sphere whose radius is a. Then the equations of motion are **wrong due to sign error because of funny definition of azimuth**

$$\frac{d\phi}{dt} = \frac{V\cos(A-r)}{a} \tag{4}$$

$$\frac{dL}{dt} = \frac{V\sin(A-r)}{a\cos\theta} \tag{5}$$

where we solve for A using Equation 3. We use a simple model of SOFIA's activities when planning flights. The observatory can be on the ground, taking off, landing, performing a turn, performing a flight leg, or performing a dead leg. A *flight leg* is a period of time during which an observation is taking place. The aircraft's final position is determined by solving Equations 4 and 5. We assume that the target remains fixed throughout the flight leg. A *dead leg* is a leg during which no observation is taking place. Dead legs are needed to reposition the aircraft in order to enable observations. The aircraft's final position is found by assuming that the ground track is a Great Circle on the surface of the sphere whose distance and heading are determined by the planner. Thus, the planning problem requires making the following choices:

- Which observations to perform.
- The order in which the observations are performed.
- The takeoff time.
- Whether or not dead legs are performed.
- Dead leg duration and heading.

Algorithm Description

In this section we describe the search algorithm we developed to solve the SFPP. We first motivate our design, then describe the algorithm and heuristics in detail.

Choice of Search Algorithm

If we look at the description of the SFPP domain, we can get an initial guess as to the search space. Notice that there are no constraints on the choice of dead leg heading or duration. If time were modeled as a continuous quantity, then the search space would be mathematically infinite; even if time were modeled as an integer, the search space would be quite large. The same issue applies to selecting the heading of dead legs. Without considering these problems, the search space is "merely" combinatorial: let N be the number of observations. Then the search space is

$$\sum_{K=1}^{N} \binom{N}{K} K!2^K$$

[2] the direction the telescope points relative to the direction of motion of the aircraft

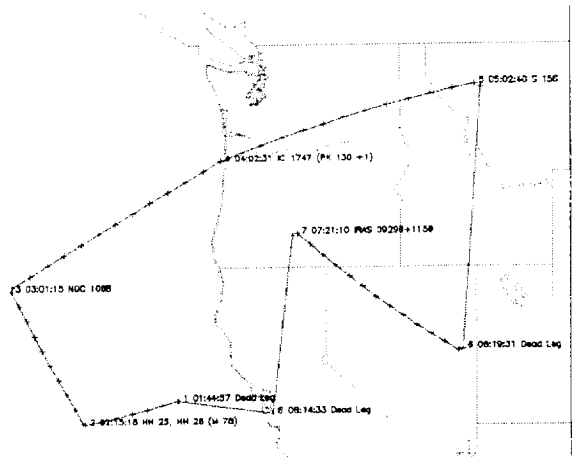

Figure 2: A flight plan.

This covers the selection of K observations from N to be present in the plan, the possible permutations of each of these observations in the plan, and finally the choice of whether or not an observation is in the plan. It is clear that as the number of observations grows large, the search space also grows large. For these reasons, we omit complete search algorithms such as branch-and-bound from consideration; these algorithms are unlikely to find reasonable plans.

Furthermore, we observe that it is necessary to know both *where* an observation begins and *when* an observation begins in order to know where the aircraft is after the observation ends. This can be seen due to the fact that Equations 4 and 5 describe an initial value problem; we must compute the azimuth A given the position of the aircraft and the time the observation is performed in order to determine the aircraft's movement. At the present, we solve these constraints via Euler's method (Ferziger 1981). It is also necessary to know where and when the observation begins to determine whether or not the observation is feasible, since this requires ensuring that the elevation constraints are not violated during the observation. While these considerations do not eliminate local search algorithms from consideration, we chose a simpler algorithm to begin with.

Since we need to know the location of the aircraft and the time of the aircraft in order to evaluate candidate flight legs, we began with a progression planning algorithm that simulates the flight from takeoff to landing. The algorithm proceeds by evaluating the observations to perform next using a combination of lookahead and heuristics that assess

Lookahead(O,K,P)
K is the lookahead distance, P is the plan
repeat K times
  $Q = \emptyset$
  while N is not empty
    for each unscheduled observation $o \in O$
      if Feasible(o,P)
        $Q = Q \cup$ (o,Evaluate($P||o$))
    if Q not empty
      o' =Select(Q)
      $P = P||o'$
      remove o' from O
    else break
return P **end**

Figure 3: A sketch of the Lookahead phase of the SFPP Flight Planning Algorithm.

flight plans. The algorithm can potentially perform a large number of flight leg construction steps, as well as searching for dead legs to enable observations (more on this in Section . To minimize these costs, instead of using a full lookahead, we use the heuristic to guide the selection of observations used in the lookahead phase as well as to commit to the next observation in the plan. The heuristic evaluations are used as the input to a stochastic selection algorithm. This results in a planning algorithm that is simple, fast, and can trade off between exploration and exploitation by virtue of how biased the stochastic selection is towards greedily selecting the best observation. The search algorithm is defined in Figure 4.

The complexity in the lookahead phase stems from calls to the function Feasible(o, P). This function ensures that an observation does not violate any of the constraints by constructing a dead leg if necessary, and constructing the flight leg to check visibility constraints. It also constructs a dead leg to the landing airport to ensure that the aircraft is not too far away after performing the observation. The routine Select(Q) uses the heuristic evaluations of each candidate extension to the plan; this routine can use any strategy ranging from purely greedy to stochastic functions that ignore the heuristics.

The overall algorithm uses the previously defined function Lookahead() in two ways. The first is to select a good takeoff time from the range of possible times. The second is to evaluate the impact of adding each remaining feasibleobservation to the plan. The Lookahead() function heuristically constructs a short flight plan that extends the existing plan, then uses a variant on the same heuristic function to evaluate each of these plans. The Select() function is then used to choose which observation will extend the plan.

We now analyze the complexity of the algorithm we have defined. We will analyze the complexity in terms of calls to Feasible(), which comprises some number of flight leg and dead leg construction steps. Let us assume that the start time has been selected. Let N be the number of observation requests, let K be the number of observations used in

ForwardPlan(K,O,H,S,P, D)
K is the lookahead distance, N is the set of observations
S is the number of start time samples, P is the plan
H is the set of possible takeoff times, D is the flight duration
$Q = \emptyset$; $P' = \emptyset$
repeat S times
  choose a value $h \in H$
  $P' = P'||h$
  P'=Lookahead(O,K,P')
  $Q = Q \cup$ (h,Evaluate(P'))
h=Select(Q)
$P = P||h$
while O not empty
  $Q = \emptyset$; $P' = \emptyset$
  for each unscheduled observation $o \in O$
    if Feasible(o,P)
      $P' = P||o$ # o includes dead leg, if necessary
      P'=Lookahead(K,O,P')
      $Q = Q \cup$ (o,Evaluate(P'))
  if Q not empty
    o' =Select(Q)
    $P = P||o'$
    remove o' from O
  else
    $O = \emptyset$
    break
return P
end

Figure 4: A sketch of the SFPP Flight Planning Algorithm.

the lookahead step, and let M be the maximum number of observations that can be in any flight plan. Then the algorithm takes $O(N^2KM)$ flight leg and dead leg construction steps. The proof is as follows: For each call to Lookahead(), we construct a plan with at most K steps, in which we call Feasible() for each of the observations not scheduled during the lookahead phase. This number is bounded above by N. Thus, each call to Lookahead costs $O(NK)$ calls to Feasible(). A plan is assumed to have at most M steps. For each of these steps, we call Lookahead at most N times, since we must consider extending the plan by each unscheduled observation. In addition, for each of these steps, we make at most N calls to Feasible() to decide which observations must be evaluated. Thus, we have $O(N^2KM)$ calls to Feasible(). The cost of evaluating S start times is S calls to the Lookahead() function, which is dominated by the rest of the algorithm.

Constructing Dead Legs

The feasibility check Feasible(o, P) attempts to construct a flight leg for observation o after executing the flight plan P. This consists of determining whether or not the visibility constraints are violated during the flight leg, and ensuring that the aircraft can fly to the landing airport after the flight leg is finished. If there is insufficient time to both perform the observation and fly to the landing airport, then the observation is considered infeasible. However, if the visibility constraints are violated, it may be possible to construct a *dead leg* to reposition the aircraft or delay the observation. For example, if the object is just below the horizon and is rising, either flying towards the object or delaying the observation by a short time can make the observation feasible.

In a perfect world, the goal is to find the shortest possible dead leg that enables a particular observation given the aircraft's position and the time. Unfortunately, it is difficult to pose this problem as a closed-form mathematical optimization problem. Thus, the Feasible() check conducts a search over candidate dead legs that may enable the observation. The search space for this consists of all possible headings and dead leg durations, which is very large. We restrict this search by limiting the maximum dead leg duration, constraining the dead legs to be multiples of a constant value, and restricting the possible heading changes to be multiples of a constant value. These values are parameters to the flight planning algorithm.

Originally, we began searching for dead legs by iterating over possible dead legs in order of their duration. This iteration ceased as soon as a dead leg was found that enabled the observation. However, we made the search more efficient by first determining the minimum change in *latitude* (subject to restrictions on the allowed dead leg durations) needed to enable the observation. This is done by conducting a linear search over dead leg durations and restricting the headings to either due North or due South. A shorter dead leg enabling the observation might exist; for instance, if the object is below the minimum elevation and rising, flying towards it will minimize the time until it is visible. So we attempt to find a dead leg of shorter duration. We do not need to search dead legs that change the latitude in the reverse direction, as

Feasible(o, P)
D is the maximum dead leg duration
#I is the dead leg increment, E is the heading increment
Construct flight leg for o
Construct dead leg to landing airport
if the flight duration and object visibility constraints are satisfied
  return true
d'= duration of dead leg enabling o by changing latitude
e'= heading of dead leg enabling o by changing latitude
if $d' > D$
  return false
for $d = d'$ to 0 by $-I$
  noDeadLeg=**true**
# Search headings +/- 90 degrees from latitude changing heading
  for $e = e' - 90$ to $e' + 90$ by E
    Fly dead leg specified by d, e
    Construct flight leg for o
    Construct dead leg to landing airport
    if flight duration and object visibility constraints are satisfied
      noDeadLeg=**false**
      feasible = **true**
      deadLeg= (d, e)
      break
  if noDeadLeg==**true**
    break
if noDeadLeg==**true and** feasible == **true**
  o = deadLeg $||o$
  return true

they are guaranteed to be longer than the dead leg we just discovered. As soon as we find a dead leg duration which could not enable the observation, the procedure halts. Note that this procedure neither finds the shortest dead leg, nor even guarantees to find a dead leg if one exists, because of the discretization of the search space.

As can be seen from the sketch of this procedure, each call to Feasible() results in at most $\frac{2D}{I} + \frac{180D}{EI}$ constructions of flight legs, enabling dead legs, and dead legs to return the aircraft to the landing airport. However, in many cases the actual number may be much smaller.

Heuristics

Heuristics play two roles in the SFPP algorithm described above. They are used to decide which observations are added during the lookahead phase, and they are used to commit to the next observation in the plan based on the results of the results of lookahead. As we have stated previously, the evaluation criteria are identical, and can be viewed as a mapping from a flight plan P to a real number. In this section, we describe the heuristics in detail.

A good flight plan is one that observes many high valued observation requests, but takes as little time as possible. Indirectly, this means minimizing the amount of dead leg time. A heuristic should also pay attention to how much time is needed to get the aircraft to the landing airport; if observations naturally carry the aircraft towards the landing

airport, that may reduce dead leg time to land after observations are completed. However, plans that "loiter" over the landing airport may not enable the most important observations. A final point is that aircraft turns cost some time **more defense of why this is needed.**

We have identified four *features* of a flight plan that serve as the input to heuristics:

1. Importance: the summed priority of the observations performed in the flight plan
2. Efficiency: the summed duration of flight legs divided by the total flight duration
3. Dead leg home distance: the amount of time required to fly to the landing airport
4. Turn amount: the total degrees of heading changes between flight legs

Some of these features work in opposition to each other. Relying on importance more than efficiency may lead to poor choices when a high priority observation is very inefficient. Relying on efficiency more than importance may lead to poor choices when high priority observations do not lead to great inefficiency. For this reason, we associate with each of these features a real valued weight between 0 and 1. Our heuristic then maps a flight plan into a real number by summing the weighted value of each feature. We can thus express a number of heuristics that play these features off each other in an attempt to identify a good heuristic.

The heuristic we describe is used in two different contexts. In the call to Lookahead(), we use the heuristic to build the best plan possible plan conditioned on an action, and intend only to use the resulting plan to assess the value of that action. In the main routine of the planner, we use the heuristic to assess the different plans we constructed in the lookahead phase, and choose the next action based on the values of the different plans. Thus, we distinguish between the weights used for the heuristic depending on the context in which the heuristic is used.

Implementation Details

We modeled the SFPP using the Extensible Universal Remote Operations Plannning Architecture (EUROPA), developed at NASA Ames Research Center (Frank & Jónsson 2003). EUROPA is a constraint-based planning system (Smith, Frank, & Jónsson 2000) that enables the modeling of complex planning and scheduling domains. Planning domains are described in terms of *attributes* representing particular aspects of the world that evolve over time, and *intervals* describing the state of an attribute over a contiguous time period. The simple model of aircraft activities described in section is easily transcribed as a EUROPA model. EUROPA also provides an object-oriented mechanism for representing partially completed plans. This mechanism represents plans as Dynamic Constraint Satisfaction Problems (DCSPs) (Jónsson & Frank 2000) and provides automatic constraint enforcement during planning. EUROPA

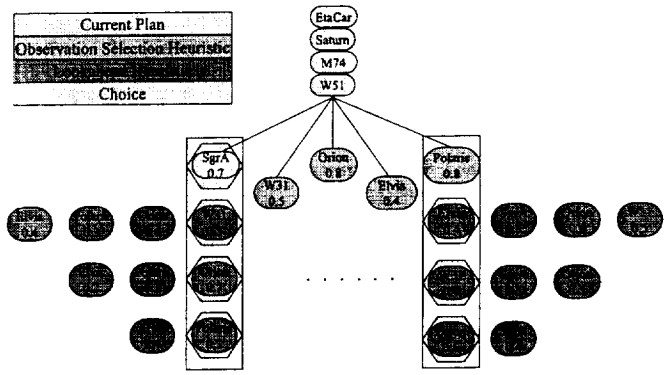

Figure 5: A pictoral representation of the search process.

also provides mechanisms for incorporating domain specific constraints such as the elevation and flight dynamics constraints present in the SFPP. The mechanism also provides an API that allows many different kinds of search algorithms to be constructed on top of it, thus providing considerable flexibility. We implemented the algorithm described in section on top of the API provided by EUROPA.

Analysis

In this section, we analyze the performance of the flight planner algorithm. We first describe the source of our problem instances. We then describe some preliminary experiments to determine good parameter settings for the algorithm parameters. We then describe experiments to determine the effectiveness of the algorithm using the parameter values we felt were most suitable on two different classes of flight planning problems.

Sample Problems

The SOFIA observatory is the successor to the Kuiper Airborne Observatory (KAO), which performed infrared astronomical observations between 1974 and 1995. NASA Ames Research Center has many years of archived flight plans that were executed aboard KAO. We obtained from KAO astronomers a set of flights that they felt were representative of the flights that will be performed on SOFIA. These flights were flown from Moffett Field, CA; Honolulu, HI and Christchurch, NZ between 1988 and 1995, and occurred between the months of April and November. The archived flight plan data contains only a specification of the flight plan that was actually executed. It contains no indication of what set of requests were considered when making the plan. Furthermore, the flight plans reflect considerations such as restricted airspace and predicted winds that our planner cannot yet handle. Nevertheless, they provide a reason-

Index	1	2	3	4	5	6	7	8	9	10	11
Airport	H	H	H	H	M	M	M	M	M	M	M
# Obs	9	9	10	10	7	8	8	6	10	8	8
Dur	437	432	440	441	460	495	500	515	610	470	600
Index	13	14	15	16	17	18	19	20	21	22	23
Airport	M	M	M	M	M	M	M	M	M	M	M
Obs	11	10	8	9	10	8	8	8	9	9	6
Dur	432	437	445	439	440	449	448	441	440	442	293
Index	25	26	27	28	29	30	31	32	33	34	35
Airport	M	M	M	M	M	M	M	M	M	M	M
Obs	7	4	7	6	7	9	8	11	10	8	7
Dur	440	316	443	440	443	451	442	443	437	448	438
Index	37	38	39	40	41	42	43	44	45	46	47
Airport	M	M	M	M	M	M	M	N	N	N	N
Obs	7	3	9	8	8	8	4	10	8	8	8
Dur	385	232	447	442	441	441	192	495	470	460	465

Figure 6: Characteristics of Single Day Instances.

12 Index	1	2	3	4	5	6	7	8	9	10	11	12
MAirport	M	M	M	M	M	M	M	M	M	M	M	M
6 # Obs	23	23	23	32	32	32	32	27	27	27	61	61
390Index	13	14	15	16	17	18	19	20	21	22	23	24
24Airport	M	M	M	M	M	M	M	M	M	M	M	M
MH Obs	61	61	61	61	61	15	15	15	15	15	44	44
8 Index	25	26	27	28	29	30	31	32	33	34	35	36
435Airport	M	M	M	M	M	M	M	M	M	M	M	N
36 Obs	44	44	44	58	58	58	58	58	58	58	58	24
M Index	37	38	39	40	41	42						
7Airport	N	N	H	H	H	H						
380 Obs	24	24	38	38	38	38						
48												
N												
10												

Figure 7: Characteristics of one-month duration Multiple Day instances.

able benchmark to compare the performance of our planner. For this reason, we created two sets of problem instances using the archived flight plans.

The first set of problem instances were designed to test the basic performance of the algorithm. These problem instances were created using a single archived flight as the basis of each instance. As such, we refer to them as the Single Day Instances. Each observations performed in the archived flight was converted into a request to observe the object for the same amount of time it was observed in the archived plan. We gave all observation requests the same priority [3]. The flight was requested to take off and land at the same airports used in the archived plan. The flight was requested to take less than 1.1 times as long as the archived flight. Finally, the flight was requested to take off no earlier than 30 minutes before the archived takeoff time, and land no late than 30 minutes after the archived landing time. The result is a problem instance for which there is a flight that enables all of the observations, but that solution has been obscured by relaxing the data in the archived plan. Furthermore, since the instance does not specify either winds or restricted flight zones, it is possible that shorter plans than the archived ones could be found.

The table below lists some salient characteristics of the Single Day Instances. We tabulate the number of observations, the archived flight duration, and the airport. In only one case did the takeoff and landing airports differ; in this case, the flight plan began at Moffett Field and ended at Honolulu. The instance numbers in this table will be used to present results of our algorithm later in the paper. **tables too large, any way to shrink them? Also, problem in that all tables indexed 1-?? Thus, in the graphs, little way to distinguish a problem instance except to read the graph caption. Perhaps add something to the problem indices like S-1 for single-day instance 1, M-2-1 to indicate Multi day instance, 2 month, instance 1? This gets messy fast as well as requiring regeneration of all of the graphs. Also, better to present the flight efficiency for these flights rather than the number of observations? DO we need both?**

[3]The KAO astronomers had indicated informal object priorities for only a subset of the flight plans delivered to us, and had no basis to compare two flight plans with different objects except for total observing time.

Index	1	2	3	4	5	6	7	8	9	10	11	12
Airport	M	M	M	M	M	M	M	M	M	M	M	M
# Obs	76	76	76	76	76	76	76	76	76	55	55	55
Index	13	14	15	16	17	18	19	20	21	22	23	24
Airport	M	M	M	M	M	M	M	M	M	M	M	M
Obs	55	55	55	55	102	102	102	102	102	102	102	102
Index	25	26	27	28	29	30	31	32	33	34	35	36
Airport	M	M	M	M	M	M	M	M	M	M	M	M
Obs	102	102	102	102	102	62	62	62	62	62	62	62
Index	37	38										
Airport	M	M										
Obs	62	62										

Figure 8: Characteristics of two-month Multiple Day Instances.

The second set of problem instances was designed to push the planner by ensuring that not all observations could be scheduled. These instances were constructed from the first set by merging requests for different flights. Thus, we refer to them as the Multiple Day Instances. To ensure that observations were still possible, we merged flights that were originally executed within one or two months of each other during the same calendar year. With one exception, we only merged flights using the same airports. We adjusted the flight duration to be the maximum of the flight durations of the old problems used to construct the new one. In addition, we adjusted the earliest takeoff time to be the minimum of the earliest takeoff times of the old problems, and the latest landing time to be the maximum of the latest landing times of the old problems. We then generated one instance with the set of all of the requests and the new duration, takeoff and landing times, for each date corresponding to the contributing Single Day Instances. Due to the variability in the number of observations in the Single Day Instances and number of flights taking place within a short period of time, the number of observations in these larger problems varies widely. In the following table we describe Multiple Day Instances generated by combining problem Single Day Instances flown within one month of each other.

In this table we describe Multiple Day Instances generated by combining Single Day Instances flown within 2 months of each other.

which instances are these? One set of flights in August of 1995 flew observations from Moffett Field, CA, and subsequently from Honolulu, HI. The Multiple Day Instances was extended to merge flight plans from both of these locations. This set of problems is somewhat different from the others, in that we expect that some of the observations might be somewhat more difficult to schedule. For instance, an

observation that could be performed over Hawaii might be difficult or impossible to schedule when flying out of California. Thus, we consider this subset of problems to be especially interesting.

It is worth noting some factors about our test problems. While the problem description admits varying observation priorities, our test problem instances all had identical priorities for observations. While the problem description allows for constraints on the legal times to perform an observation, our problems have no additional constraints beyond those imposed by the takeoff and landing time. Finally, the problems have known solutions, and thus might be considered "easy" for this reason. Our problem formulation strategy ensures that these solutions are disguised by increasing the search space in a manner consistent with our expectations on the actual flight planning problems.

Parameter Settings

We conducted preliminary experiments on a small set of problems from the Single Day Instances in order to choose good parameter values for the algorithm. We chose one flight each from Moffett Field, CA, Christchurch, NZ and Honolulu, HI to perform our initial experiments.

We considered the following parameters:

- Whether to include or omit each feature of the heuristic when called from Lookahead.
- Whether to include or omit each feature of the heuristic when called from ForwardPlanner.
- Lookahead value.
- Number of start times.
- The proportion of the time the heuristic is used to drive greedy selection as opposed to using the output of the heuristic probabilistically. We allowed different proportions for the Select function calls in Lookahead and in the main algorithm.

Initially, we did not vary the parameters governing the search for dead legs. For these experiments, ee restricted the maximum dead leg duration to 4 hours, the dead leg increment to 1 minute, and the heading increment to 7.5 degrees. We describe experiments varying the dead leg search parameters in section .

We searched for combinations of parameter values that ensured good performance on the subset of problems we used for these experiments. Good performance in this case meant that the parameter settings enabled the algorithm to find schedules in which all of the observations were performed, and the overall flight duration came as close as possible to the duration of the flight executed in the KAO archives.

Due to the large space of variations in the algorithm settings, we investigated only a small fraction of the possible combinations. Our results led to the following informal conclusions, in rough order of importance:

1. Employing a suitable lookahead distance was important to achieving good performance. A lookahead distance of 4 observations was sufficient to achieve good performance.

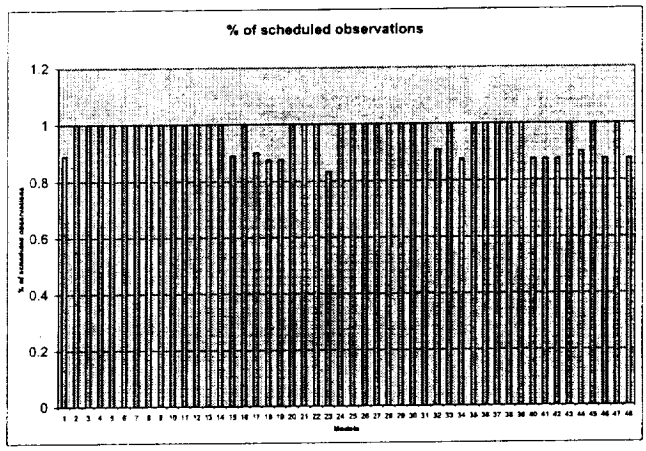

Figure 9: The percentage of Single Day Instances for which all observations were scheduled.

2. Using the heuristics greedily was always better than including any amount of randomization, both during and after the lookahead phase.
3. Examining 5 start times was sufficient to achieve good performance.
4. When evaluating observations in the lookahead phase, evenly weighting all 4 features of the heuristic achieved good performance. We did not analyze uneven weighting of the features.
5. When evaluating observations after lookahead, only Importance and Efficiency were analyzed. When weighted evenly, the algorithm achieved good performance.

These settings were then used in the experiments that are described in the following sections.

Single Day Instances

After settling on the parameters of the algorithm, we ran the flight planning algorithm with these parameter settings on all of the Single Day Instances. The goal of this experiment was to ensure that the parameter settings we found were not biased by the small number of problem instances. Again, our goal was to ensure that the planner found plans that enabled all of the observations, and whose flight duration was close to that of the plans in the archive.

Figure 9 shows our results. We see that for most of the problems, we were able to schedule all of the observations. In those cases when the algorithm could not schedule all of the requested observations, only one or two observations were omitted; this is shown by the results in Figure 10.

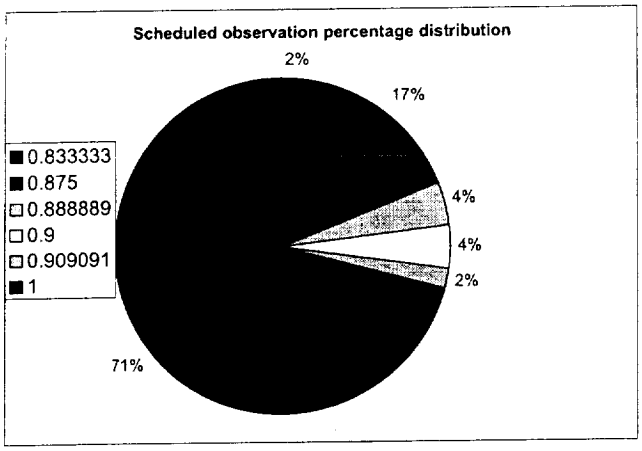

Figure 10: The maximum percentage of observations scheduled for each problem Single Day Instance.

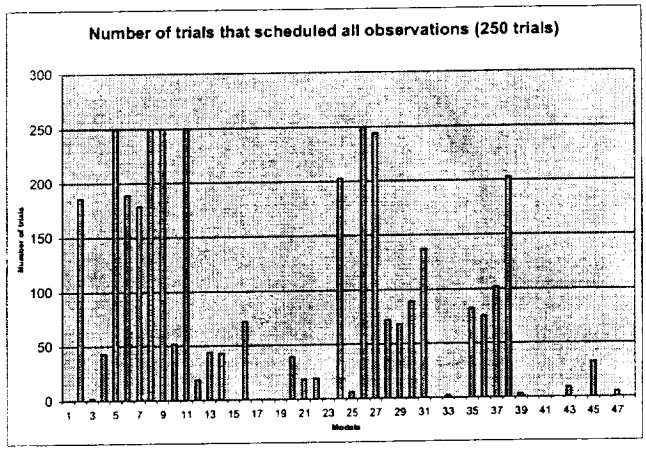

Figure 11: The percentage of time all observations were scheduled by the flight planner.

Figure 11 shows how often the planner found plans scheduling all of the observations. These results indicate that problem instance difficulty varied widely. In 5 of the 48 instances, the algorithm was able to schedule all observations every time. However, in 6 of the 48 cases in which the algorithm found a schedule including all observations, fewer than 10 tries of 250 successfully scheduled all of the observations.

Recall that the Single Day Instances limit the duration of the flight to be 1.1 times as long as the archived flight plan. We wanted to compare the duration of flights found by the planner that scheduled all of the observations to the archived flight durations. Figure 13 shows differences in the flight duration of the shortest flight for which all observations were scheduled and the flight duration of the flight in the archives. From this data we see that the planner can find plans that take roughly the same amount of time that the archived plans take and still schedule all of the observations. There are some exceptions; in three cases, the shortest flight found by the planner was more than 20 minutes longer than the archived flight. Overall, however, the results are quite encouraging.

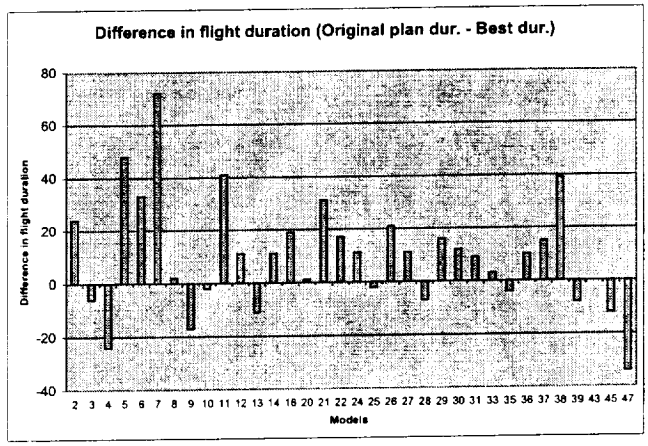

Figure 12: Difference in flight duration between archived plan and shortest plan found in which all observations scheduled. The x axis shows the duration of the plan in the archive minus the duration of the best plan found by the planner. Problem instances for which the planner did not find a plan scheduling all observations are omitted.

While the *shortest* flight plans that schedule all of the observations look good, what is the *average* duration of a flight plan that schedules all of the observations? Figure 12 shows the difference in the average flight duration of all flights found by the planner that enabled all of the observations and the duration of the archived flight plan. The news is somewhat worse here, in that the average flight duration is

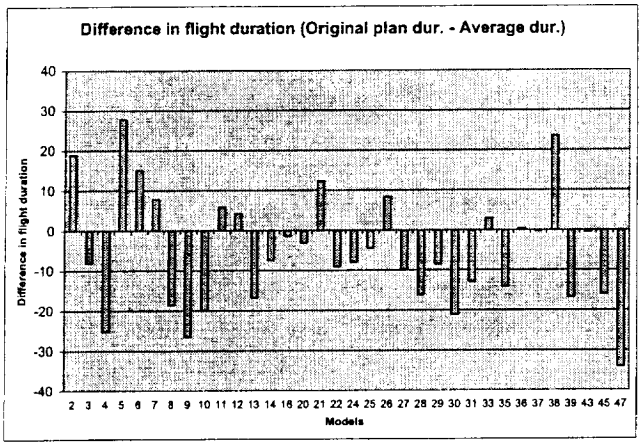

Figure 13: Difference in flight duration between archived plan and average of all plans in which all observations scheduled. The x axis shows the duration of the plan in the archive minus the duration of the average duration of plans including all observations. Problem instances for which the planner did not find a plan scheduling all observations are omitted.

frequently as much as 20 minutes longer than the archived flight duration, and in one case is 34 minutes longer than the archived flight. However, the news is generally good; the planner generally finds short flight plan with all of the observations scheduled. Note that these results should be taken with a grain of salt, because the archived flights account for restricted airspace, which tends to lengthen flights, and nominal winds, which can either lengthen or shorten a flight. Thus, nothing should be read into the precise differences in flight duration.

Multiple Day Instances

Some work required to generate the graph I want for this We then ran the algorithm with the same parameter settings on the Multiple Day Instances. Since no archived flight matches the Multiple Day Instances we have constructed, we cannot compare the performance of the algorithm on these instances to previously generated plans. Our chosen metric for measuring performance is the *flight efficiency*, which is the percentage of the flight spent performing observations.

Figure ?? shows the performance of the algorithm on the instances described in Figure 7.

the number of observations scheduled and the shortest duration of the flights. Figure ?? shows the performance of the algorithm on the instances described in Figure 8.

We see that the algorithm we have developed is able to find a good flight plan even when there are too many observations to fit into a single flight.

Figure ?? shows the performance of the algorithm on the Multiple Day instances in terms of the number of flight plans found that maximized the number of observations scheduled.

Is it worth showing this? Figure ?? shows the *Pareto Frontier* for some flight. In this figure, we plot the flight duration versus the number of observations scheduled. The Pareto Frontier is the set of all flights for which no flight is strictly better. In this case, a flight is strictly better than another if either it has the same flight duration but schedules more observations, or makes the same number of observations and takes less time. This figure is typical of the results we see for the output of flight planning.

Given that the Multiple Day Instances are composed of Single Day instances, and given that we know that all of the observations *can* be scheduled, we analyzed the results of the flight planning to determine how often each observation *was* scheduled. We wanted to see if the flight planning algorithm or heuristics led to some biased sampling of the set of the solution space. To perform this analysis, we looked at every flight plan generated by the algorithm for some problem instances.

Figure ?? shows the proportion of the observations that were not scheduled in any schedule found by the planner. We see that for some problems this proportion is quite high. This indicates that, for whatever reason, there is considerable bias in the schedules that are actually found. In some cases we would expect this, for example, for the multi-day schedule including Moffett and Hawaii requests. In other cases, this may be an indicator of some sort of actual bias **check to see if the cases where we get low percentages are composed of single day requests far apart in the calendar, etc.**

Tuning Algorithm Performance

In conducting our experiments on the Multiple Day Instances, we discovered that algorithm computation time became quite large. The main reason for this was the dead leg search. As we saw in section , the algorithm may perform a very large number of elementary operations during the search for a short dead leg. The maximum dead leg duration D, dead leg increment E and heading increment I can all be used to limit the number of prospective dead legs constructed. However, reducing these prospective dead legs may come at a cost in the value of the resulting flight plans. Reducing the maximum dead leg duration to zero may lead to poor plans because some observations may not be possible. Similarly, reducing the dead leg or heading granularity may lead to inefficient plans.

Figure ?? shows the impact of changing D and I on a problem instances from Moffett; this is instance 19 from the instances in Figure 8. For each of the parameter settings we ran 100 samples of the flight planning algorithm. The figure shows the number of observations scheduled in the best flight plan found during search. All other algorithm parameters were otherwise the same. We see from this figure that modifying the dead leg search algorithm had little impact on the number of observations scheduled in the best plans found.

Figure ?? shows the planner speed as a function of the dead leg search parameter values. We see that the planner speed increases as the number of candidate dead legs is reduced.

Conclusions and Future Work

We have described a telescope observation scheduling problem motivated by the SOFIA General Investigator program. The problem is unique when compared to other scheduling problems in that it involves complex, sequence-dependence constraints governing the simultaneous apparent motion of the objects and the observatory. These are complicated by the requirement that the observatory return to a designated location. We have described a forward search algorithm and associated heuristics, and demonstrated that the resulting search algorithm performs well on a realistic benchmark problem.

The problem described in this paper makes a number of simplifying assumptions. The planner does not account for restricted airspace, which can influence the order of the observations and the characteristics of dead legs. The constraints that implement the equations of motion for the aircraft currently ignore wind direction and velocity. In addition, astronomers may impose additional requests on observations, such as a minimum observing altitude. Finally, the flight duration constraint is a surrogate for a variety of constraints on the fuel consumption profile of the aircraft. These assumptions are being removed, and are likely to require some modifications of the planning algorithm in order to achieve good performance.

The algorithm described in this paper uses an "incomplete" lookahead phase, in that only one extension of the plan is constructed for each unscheduled observation. This can be generalized to permit the construction of a fixed number of extensions, should the need arise. However, algorithm performance does not presently indicate that this is needed.

The algorithm described in this paper assumes that the algorithm parameters remain static during the course of solving the problem. It would be interesting to consider *changing* some of these parameter values as scheduling proceeds.

Not sure how detailed we want to be here Finally, the algorithm described currently uses a progression-planning based approach to solving the problem. It is possible to define local search algorithms for solving this problem as well. These algorithms may suffer from two different kinds of problems. Because of the complex nature of the constraints, it is difficult to check for observation feasibility unless the initial position and time of the aircraft are known. If the plan is always considered "complete", poor performance due to the need to constantly recompute the state of the plan. Alternately, the plan can be considered "incomplete", which saves time performing intermediate computations, but may lead to infeasible plans that are not detected. We have formulated an alternate planning algorithm based on Squeaky Wheel Optimization that employs the Forward Planner without lookahead as a subroutine. It will be worthwhile to compare the two approaches.

Acknowledgements

We would like to thank Sean Casey, the SOFIA Facility Instruments Program Chief Scientist, for the time he has spent describing this problem. We would like to thank Sean Colgan and Allan Meyer, former KAO astronomers and members of the SOFIA staff, for helping mine the KAO archives and providing us with benchmark problems. Lastly, we would like to thank Michael Gross of USRA for his advice and assistance. This work was funded by the NASA Intelligent Systems Program and the NASA Research Technology Opportunities Program.

References

Ferziger, J. 1981. *Numerical Methods for Engineering Applications.* John Wiley and Sons.

Frank, J., and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Journal of Constraints* To Appear.

Jónsson, A., and Frank, J. 2000. A framework for dynamic constraint reasoning using procedural constraints. *Proceedings of the Euopean Conference on Artificial Intelligence.*

Meeus, J. 1991. *Astronomical Algorithms.* Willmann-Bell, Inc.

Smith, D.; Frank, J.; and Jónsson, A. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15(1).